The EXAFS Data Editor

Matthew Marcus

May 9, 2002

I. Introduction

This is the user manual for the EXAFS data editing program. This program, written in LabVIEW, provides the tools needed to go from raw data to a 2-column plot of signal vs. energy. It includes the following functions:

1. Data input. At present, it reads only the ASCII format designed for Beamline 10.3.2. This format is a simple, multi-column ASCII layout with a header, so it should be easy to convert from other formats.
2. Data output. After processing the data, the program will write either 10.3.2 multi-column output or 2-column. The latter is suitable for input into most EXAFS analysis programs.
3. Averaging. Since a single scan is rarely good enough, the program lets you 'stack' or average multiple scans. The counts and the times in each channel are added up. If the grid of energies don't match, the data are interpolated. The files included in the sum are listed. You can clear the average (remove all files).
4. Inspection. You can plot up any sum of scalers, the ratio of any sum of scalers to any other sum, or the natural log of any such quantity. The LabVIEW native graph tools let you zoom in on any part of the data. For more information about LabVIEW conventions and graph tools, see the note *Common LabVIEW Conventions*.

   You can also define a linear background which is subtracted from the displayed data. This doesn't affect the data in memory, but just lets you look at sloping data

more closely. This background can be toggled on and off. You can either define the background line by choosing two points with cursors, or you can pick a data range and let the program run a best-fit line through the data.

5. Deletion and truncation. You can set cursors on the graph and then remove the data between the cursors or the data outside the range of the cursors. The program offers two kinds of deletion. One of these is the simple removal of data points. The problem with this method is that if you're averaging files, you could make a gap in the data which isn't filled in by the other files you enter. Therefore, the default mode is to assign the 'bad' points a vanishingly small count time. Thus, if some other file being averaged in has good data in that range, the interpolation will work without trouble and the 'bad' data will be effectively ignored.

6. Glitch removal. Instead of removing points corrupted by monochromator or Bragg glitches, you can 'smooth' over them. In this sub-program, four cursors appear on the graph. Call them cursors 0 through 3 in order from left to right. A cubic polynomial is fit to the data between cursors 0 and 3, ignoring the points between cursors 1 and 2 (the inner pair, where the glitch is). A separate polynomial is used for each scaler channel. The data between cursors 2 and 3 are fit to the same polynomial as that between 0 and 1. The data between cursors 1 and 2 are replaced by the fit. This process happens interactively in that you can move the cursors around and see the replacement data change.

   There is another, similar form of glitch removal, intended for use with multi-element detectors in which one element may receive a Bragg beam from the

sample for a few eV of the scan. In this method, you define the other, un-corrupted scalers as reference scalers. The program then does a cubic fit to the ratio of the values of the glitched scalers to the reference ones. In the glitched areas, the data for the glitched scalers are replaced by the product of the fit and the sum of reference scalers. Thus, instead of fitting EXAFS wiggles, you fit a slowly varying ratio of counting efficiencies between channels.

7. Jump removal. Every so often, the beam moves or some other effect causes a discontinuous change in the sensitivity of one or more detectors. The result is that the data has a jump in one or more scalers. To fix this, you can use a variant of the glitch-removal process. You still get the four cursors, but this time, the data between cursors 0 and 3 are fit to a cubic times a step function whose value is 1 on the left side and an unknown constant on the right. This additional free parameter describes the amount of the jump. If it were 1, there would be no jump. The data past cursor 2, all the way out to the end of the data set, are then divided by that jump factor.

8. Deadtime correction. This is used with pulse-counting detectors in which a high count rate causes a decrease in counting efficiency. Typically, this detector would be one or all of the elements in the 7-element Ge detector. Suppose for instance that you are looking at one element of the detector. The counts you're interested in will be those coming in within a certain range of energies (pulse heights). To do deadtime correction, you would also record a channel which includes all counts coming into the detector element (for our detector, MCA bins 0-2047).

Now, we assume that the efficiency h of the detector drops off exponentially with count rate:

$$h = \exp(-t\, C_t^{act}) \tag{1}$$

$$C_t^{obs} = h C_t^{act} \tag{2}$$

$$C_f^{obs} = h C_f^{act} \tag{3}$$

where C_t^{act} is the actual count rate for all energies and is not measurable, while C_t^{obs} is the observed count rate. Similarly, C_f^{act} and C_f^{obs} are the actual and observed count rates in the fluorescence channel. The object of the exercise is to infer C_f^{act} from C_f^{obs} and C_t^{obs}. This is done by solving (1) and (2) for C_t^{act} and h and then substituting into (3) to solve for C_f^{act}. In order to do this, one needs to know the deadtime t, which has been empirically measured to be 3.2μs for a single detector element or 0.46μs for all seven, with the usual 1μs peaking time in the XIA electronics.

The Deadtime Correction tab allows you to specify a scaler which is considered to represent the total counts C_t^{obs} and one or more which specify C_f^{obs}. It also has a control for the deadtime.

9. Overflows. This is somewhat of an anachronism, but might be needed. Old-fashioned CAMAC scalers are 24 bits deep. If you have a fast detector or an analog signal running into a high-frequency V/F, then it is easy to overflow the counters, causing very strange-looking spectra. The EXAFS editor offers several tools for fixing these problems. You can assert that a given scaler has overflowed a given number of times and simply add that count to the scaler. This can be done

either over the whole data set or past a point defined by a cursor. You can also search for jumps which were caused by an overflow and fix them automatically.

10. Energy shift. If the monochromator drifts or its energy calibration change between data files, then you can't average the files without shifting the abscissa. In this part of the program, the assumption is made that the energy shift is really a shift of the Bragg angle of the monochromator. You tell it the energy the monochromator thought it was at and where it really was, and the program works out the angular shift and adds that to all points. This process is analogous to registering images before stacking them.
11. Change offsets. This lets you change the amount of offset (dark count rate) subtracted from each channel.

II. The main screen

When you first click on the program, you get a file-open dialog box, which prompts you to find the file you want to work on. When the file is read in, it's plotted on the data graph, which is part of the main screen shown in Figure 1. If it doesn't show up, it may simply be off-scale. To bring it back, use the `Scale` tools. Two clicks will do it. The description of what to plot is handled by the `Plot Specification` control. As a convention, I'll use `Courier` font for names of controls or icons. This is a set of 'LED's. The top row specifies which scalers are added together to form the numerator and the bottom row specifies the denominator. The big light at the top specifies whether to take the log (natural) of the result. The light at the right side of the box tells the system to forget all the above and just plot the count time. This can be useful for figuring out

how many files went into the average. Thus, if you want to plot scaler 1 alone, you turn on (by clicking on it) the second light from the left in the top row, leaving the others off. Remember, scaler numbers start at 0, so 1 is the second one. If you want to plot the log ratio of scaler 0 to scaler 1, turn on the `ln(plot)` light, the first one in the top row, and the second one in the bottom row. This plot specification language is the same as is used in the data-taker. The default plot specification is that which is given in the file header.

The two big buttons are the `Stop` and `Undo` buttons. The former has the obvious function and is preferred over using the small, stop-sign-shaped icon in the LabVIEW toolbar. The second undoes the last thing you did. If you hit it again, it redoes whatever it was.

If you want to read in another file, use the `Read new file` button between the `Stop` and `Undo` buttons. This action replaces the current data with the contents of the new file. You can Undo it, as you would any other action. The running average is preserved. File selection is done using the mechanism common to the other EXAFS-analysis programs, so the folder from which the last EXAFS data were gotten becomes the default for this program's file dialog.

The data file has a header which contains all sorts of information about the data, such as the energy intervals, the title line, the number of scans in the set, etc. This header is shown as an array of character strings below the data graph. Figure 1 shows part of that display, the rest having been cropped off in making the Figure.

The main action controls are in the `Command` tabs. As shown, only the `Glitch/Jump removal` buttons are visible. You get at the others by clicking on the

appropriate tabs. It's a graphical version of pulldown menus. The operation of each tab and the commands it exposes will be described below.

Below the `Command` tabs are buttons for averaging and displaying the average. The program maintains a running average data set which starts out empty. When you hit the Average button for the first time, the file currently loaded is copied to the average. You don't see any change, except that the `Display average` button, which had been grayed out, is now enabled. When you read in another file and hit the button again, the running average now represents the sum of the two files. When you hit the `Display average` button, the data being displayed is replaced by the running average. You can undo that with the `Undo` button. Thus, you can look at a long string of data files, edit each one, put them in the average, look at the average as you go along to see how it's doing, then write out the average file. There is also a `Clear average` button which lets you start afresh. This, too, is covered by Undo protection.

III. Glitch/jump removal

Now I will describe the specifics of the glitch/jump removal commands, which are accessible through the three buttons shown on the tab control in Figure 1. Pressing the `Glitch removal` or `Jump correct` buttons brings up a screen like that in Figure 2. The `Glitch removal using reference scalers` brings up a screen which is similar but has more space for the plot specification control. There are two reasons for this difference; glitch removal with reference scalers is implemented in a separate subroutine, and also this function tends to be used with multi-element detectors for which there will be many scaler channels.

First, consider glitch removal, as shown in Figure 2. The procedure is:

1. From the main screen, use the graph tools and the plot specification controls to zoom in on the area of interest.

2. Push the `Glitch removal` button. The screen in Figure 2 pops up with the graph abscissa limits the same as in the zoomed areas you selected in the main screen.

3. Move the crosshairs so the inner pair brackets the glitch and the outer regions cover smoothly-varying data which could plausibly be fit with a cubic. The red curve will show what will happen to the data in the inner region.

4 If you like what you see, hit the `Accept` button. If that's grayed out, there's something wrong. Perhaps there's not enough data in the outer region to do a fit. Fix the problem. You can always hit the `Cancel` button and escape.

5. You can change the plot specification from within this subsystem and it will not affect the display when you return to the main screen.

Jump correction works in the same way. In fact, the toggle marked Action Selection in Figure 2 lets you change modes without having to go back to the main screen.

For the two functions discussed above, it doesn't really matter how the plot specification is set; only the display is affected by that control. However, when using reference scalers, you have to select which scalers are to be operated on and which are the reference scalers. This is done using the plot specification control. The numerator scalers are the ones to be operated on and the denominator scalers are the references. Thus, the program fits a cubic through the curve plotted with the given choice of plot specification. It is considered an error to have any overlap between the sets of reference and glitched scalers. The routine won't let you `Accept` the changes without fixing this problem if it exists. Since there may be many scalers, the index display has been left on the numerator and denominator parts of the plot specification so you can access parts of these arrays not visible by default.

IV. Deletion and truncation

The tab for deletion and truncation is shown in Figure 3. When you engage this tab, two cursors appear. You can move them about to select the region you want to delete. When you've decided, you can push one of the two Action buttons, `Delete (==...==)` or `Truncate (...==...)`. The little ASCII graphics are reminders of what these buttons do, with the dots representing deleted data. The toggle switch selects the kind of deletion to do, as discussed above. You can delete multiple regions and then return by hitting the `Return` button.

V. File writing

This is pretty self-explanatory. There are three buttons, one for writing a 10.3.2-style multi-column file, one for writing a 2-column ASCII file based on the current plot specification, and one for writing a file suitable for HEXAFS, the EXAFS-analysis program used by the Manceau group. This format is the same as the two-column format except for the addition of a small header. If any of these buttons is pushed, a file dialog will appear allowing selection of the name of the file to write.

VI. Background

There are three buttons on this tab. None of them affect the stored data, only how the data are displayed. The `Linear between cursor points` causes a pop-up screen to appear on which there is a copy of the current graph with two cursors on it. A red line appears between the two cursors. You can manipulate that line by moving the cursors, then you can hit either `Accept` or `Cancel`. Both buttons return you to the main screen. After hitting the `Accept` button, the background line is subtracted from

the displayed data. The `Flatten (fit line to data)` works the same way except that the cursors are locked to the curve and the background line is a least-squares fit to the data between the cursors. The `Undo/Redo background` button toggles whether or not to apply the background. The background is lost whenever you read in a new file or change the plot specification.

VII. Overflows

Figure 4 shows the overflow correction tab. The top control sets the bit depth of the counter and is shown set at its default value of 24. The middle control is the one for automatic removal of jumps caused by overflows. The criterion is that the discontinuity between any two points should be made as small as possible by adding a multiple of the appropriate power of two to the counts past the discontinuity. Jumps are only detected within the limits of the current plotting area. Thus if the bit depth is 24 and two adjacent points differ by more than 2^{23} counts, a multiple of 2^{24} is added or subtracted from all data past the big jump. However, there's a problem with this algorithm. If the count time changes a lot between one point and the next, the algorithm may add the wrong number of jumps because it's the rate that we want to be continuous, not the counts. Until I fix that bug, that's where manual controls come in.

Another reason to have a manual override is that if all points have an overflow, there's no way of deducing that fact from the data. All that will show up is that the data don't normalize well. However, if you know that this has happened, you can use the controls labeled Manual override in Figure 4. Reading across, you can see that this

option adds a given number of overflows to a given scaler channel, either for all points or for all points past a cursor, which appears if the toggle is in the appropriate position.

VIII. Energy shift

This tab, shown in Figure 5, lets you shift the spectrum back and forth in energy (actually, mono Bragg angle). You tell it the energy of some point and what that energy really is, and it does the rest. You can load the 'old' energy either by typing it in or using a cursor. That cursor is enabled with the slide switch and its abscissa value loaded into the `Old energy` control by pushing the `Load from cursor` button. Hitting the `Do it!` button makes the change happen. If you're not careful, you could end up hitting the button twice, thus making twice the expected shift. If that happens, well, that's what `Undo` is for.

IX. Deadtime correction

This tab, shown in Figure 6, allows one to do deadtime correction as explained above. You pick the total-counts scaler using the digital control and pick the set of counters to correct using the lights. The deadtime defaults to 0.46μs, the appropriate value for all 7 elements, considered as one big detector. The `DoIt!` button won't be enabled until you select a set of counters to correct. Be careful not to push this button twice, thereby attempting to correct what's already been corrected.

There are times when you want to treat all 7 detector elements separately. In that case, you have to record total counts and fluorescence-channel counts for each element and do the deadtime correction separately for each. This operation can get tedious. To make this easier, there is a mechanism for reading instructions about what to do from a

file. Pushing the lower `DoIt!` button causes the program to ask for the name of a file, with a default `cfg` extension, which contains those instructions. A typical file looks like this:

```
[deadtime]
deadtime=0.46
[correct 0]
total=3
correct=2,3
[correct 1]
total=11
correct=4,11
[correct 2]
total=12
correct=5,12
[correct 3]
total=13
correct=6,13
[correct 4]
total=14
correct=7,14
[correct 5]
total=15
correct=8,15
[correct 6]
total=16
correct=9,15
[correct 7]
total=17
correct=10,17
```

The file is in the typical LabVIEW Config-file format, with sections and keys, like a Windows `ini` file. The top, [deadtime], section tells the program what the deadtime should be. The deadtime defaults to 0.46μs if this section is missing. The succeeding sections each specify one correction operation, equivalent to one push of the upper `DoIt!` button. In each section, there is a key which specifies which scaler holds the total counts, and another key specifying which scalers are to be corrected. For instance, the `[correct 5]` section says that scaler 15 is the total counts, which is to be used to

correct both scaler 8 (the fluorescence channel) and 15 (the total channel itself). The numbers assigned to the sections are there only to mark them as distinct, since the LabVIEW Config-file routines don't know what to do with sections or keys having identical names.

X. Sum scalers into scaler

Suppose you are using a multi-element detector and want to drop out the signals from some detector elements. One example of this would be if one of the detectors has a problem for that spectrum. Now, if you want to average such spectra, it would be convenient to have a channel which represents the sum of all the 'good' elements. This tab allows you to sum a set of scalers and put the sum into another scaler channel. The recipient scaler must exist. As with the Deadtime Correction tab, you specify a set of scalers to operate on, in this case the ones to be summed, and an individual scaler, in this case the one which receives the sum. You then hit the `DoIt!` button. Figure 7 shows this tab.

XI. Change Offset

This final tab is for when you realize that you didn't do a dark count and later want to add in an offset. Figure 8 shows the tab. Fill in the amounts (rates) to add to the offsets for each scaler in the array to the left, then hit the `DoIt!` button.

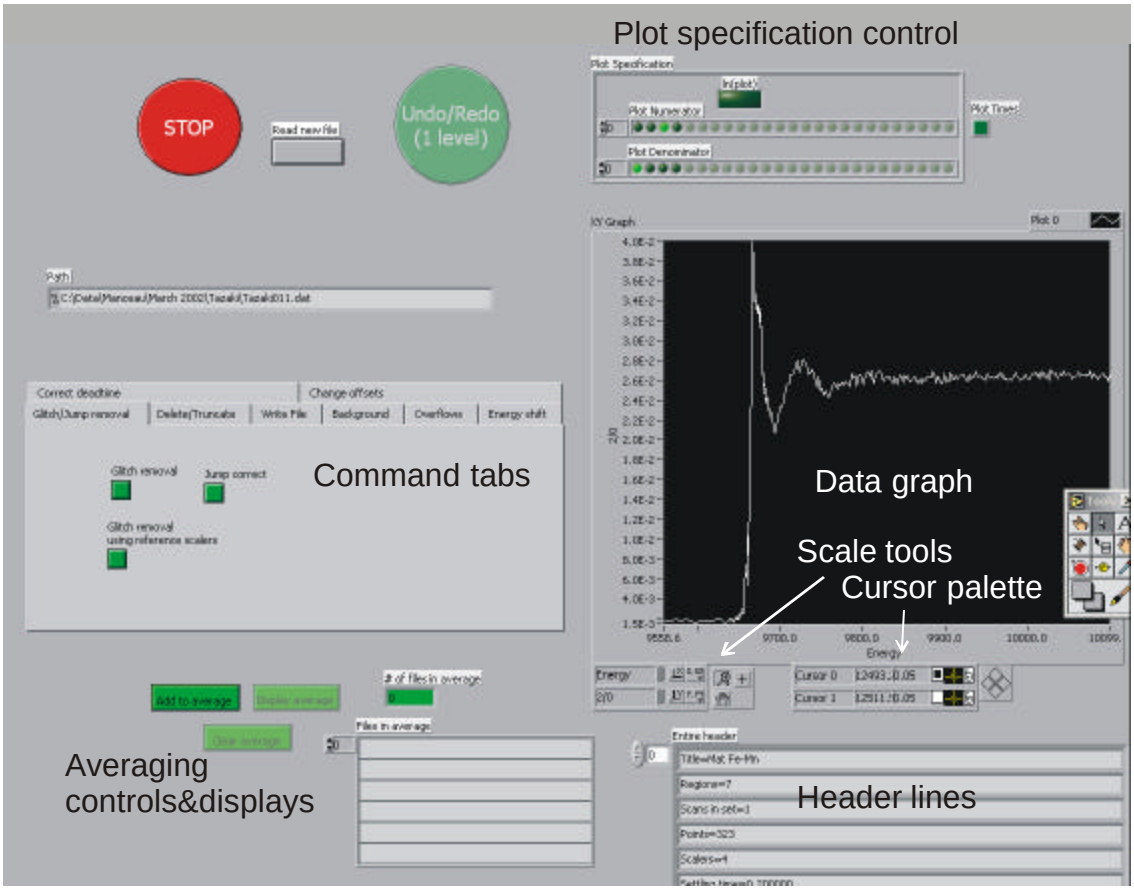

Figure 1. The main screen. This is the entry point. Items shown in Arial *are labels for this Figure and not part of the actual screen.*

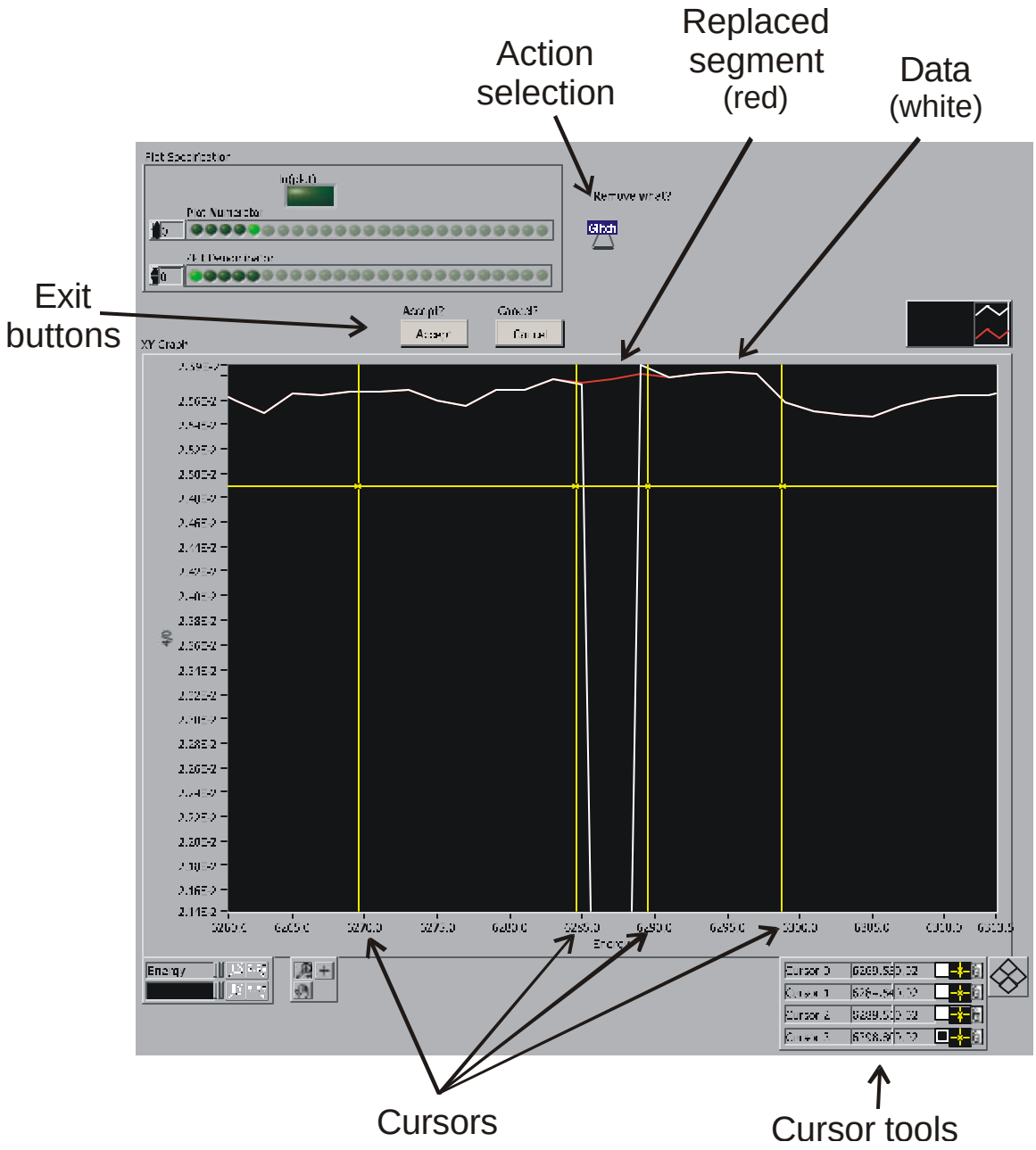

Figure 2. The glitch/jump removal screen, showing glitch removal.

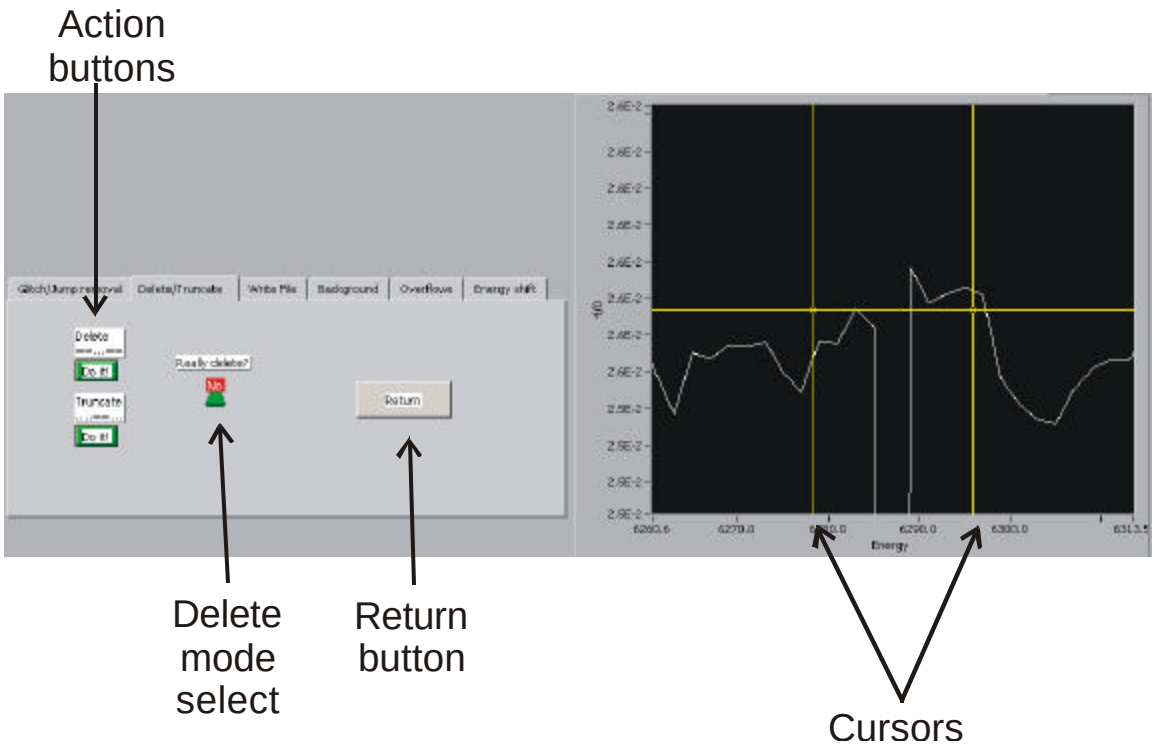

Figure 3. The delete/truncate tab and the plot with the cursors which appear when this tab is active.

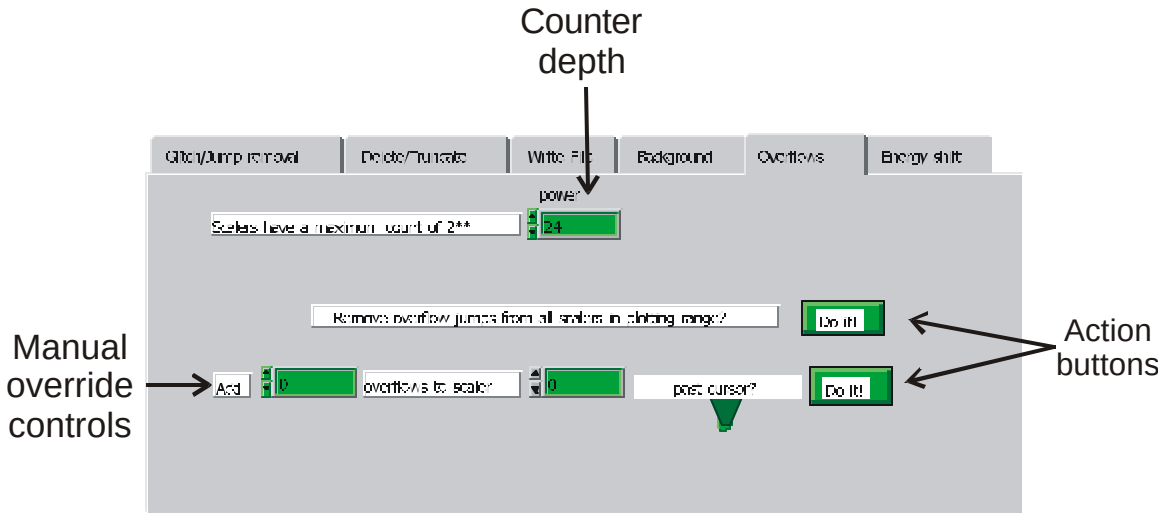

Figure 4. The overflow tab. The other setting of the toggle switch is `Whole data set`.

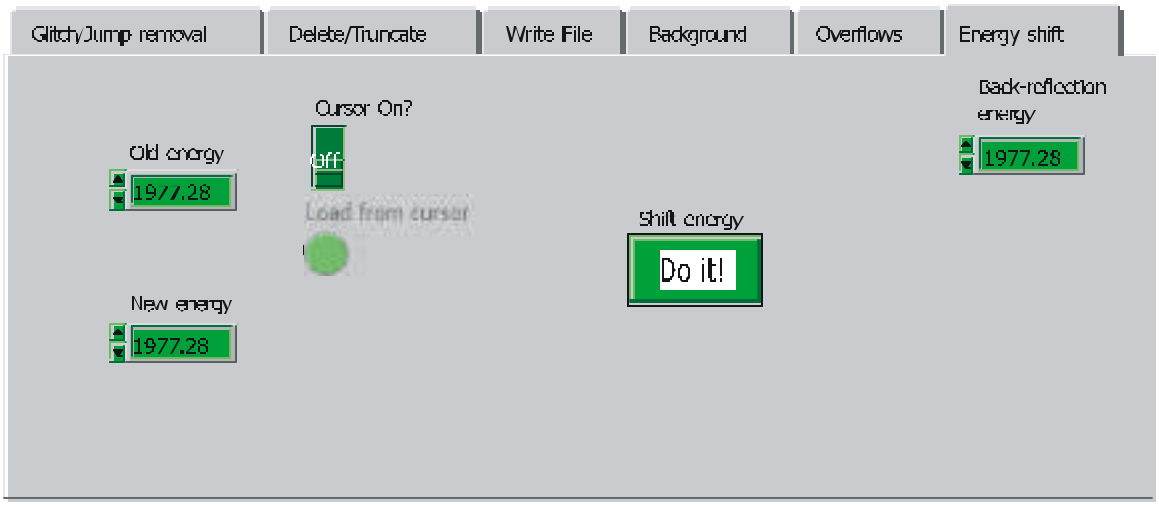

Figure 5. The energy-shift tab. The **`Load from cursor`** *button is grayed out because the* **`Cursor On?`** *switch is Off. The* **`Back-reflection energy`** *is set at its default value which is appropriate for Si(111).*

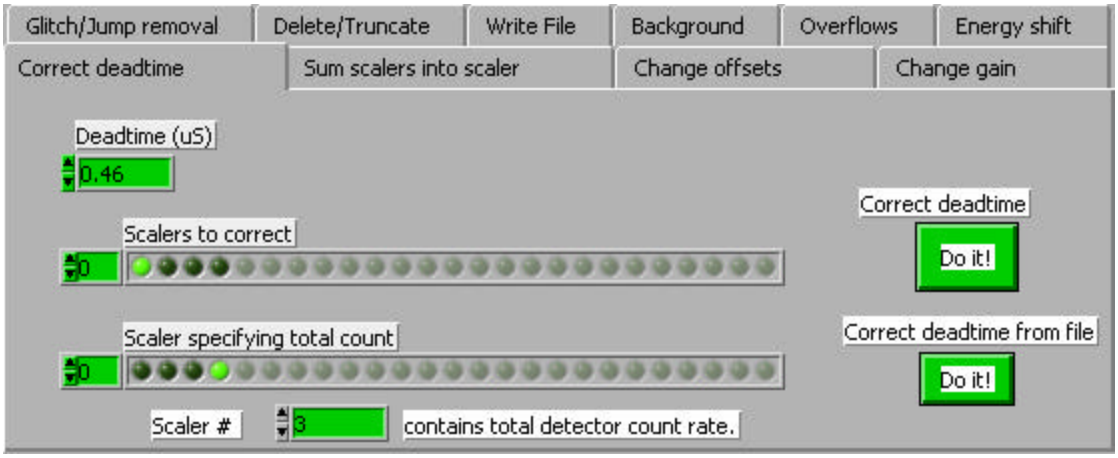

Figure 6. The deadtime-correction tab. The **`Deadtime`** *indicator is pre-loaded with the appropriate time for the whole 7 elements considered as 1. The fluorescence channel is 2 and the total-counts channel is 3. Pushing the upper* **`Do It!`** *button will cause the correction to be applied. Pushing the lower* **`Do It!`** *button will cause instructions about deadtime correction to be read from a file.*

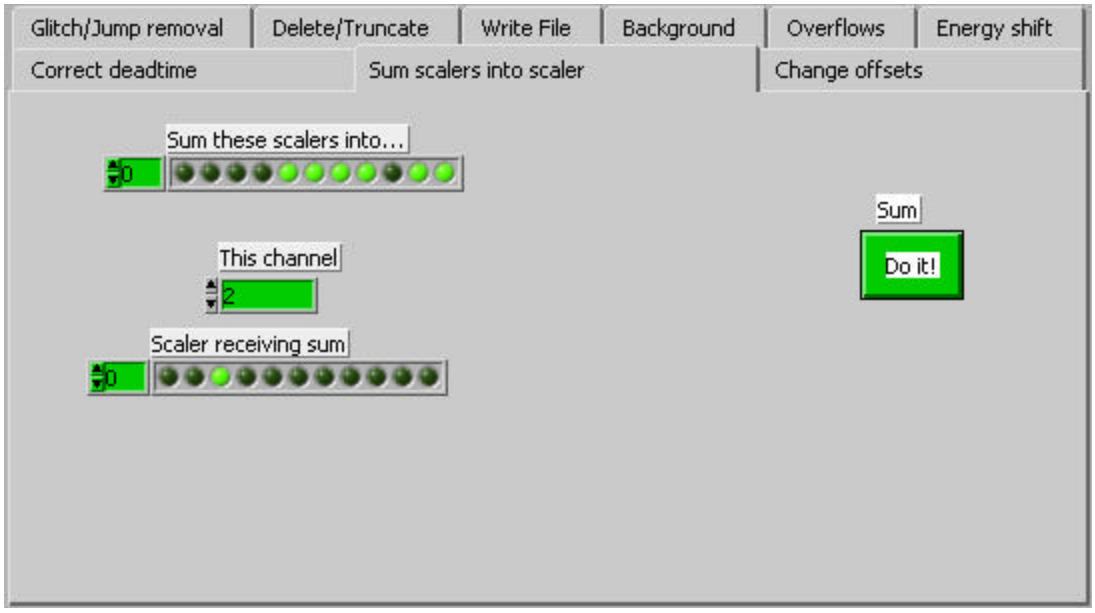

Figure 7. Sum scalers into scaler tab. This is set to add scalers 4,5,6,7,9,10 and put the sum into scaler 2, wiping out what was there.

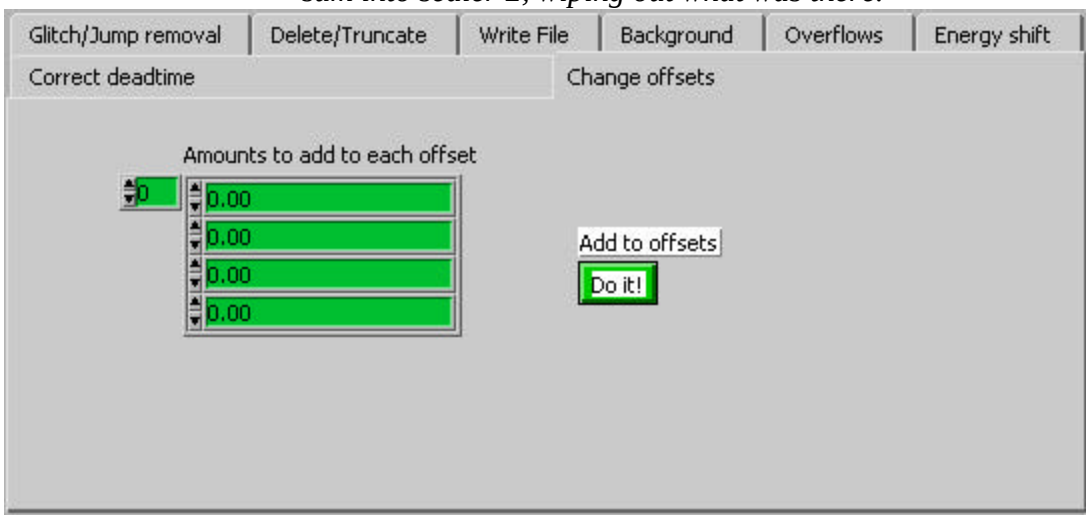

Figure 8. The change-offset tab. Fill in the amounts to add to each scaler's offset, then hit the `DoIt!` *button.*